Improving Avg Exec Time in ds-sim

## Abstract

Intelligent job dispatcher

## Introduction

Due to there are a lot of servers in the ds-sim system, and the configuration of each server is not the same. Moreover, there are many jobs that need these servers to complete, and the performance requirements of each job are also different. Therefore, for this complex situation, the optimization direction I chose is to improve the performance of each server to complete the work that reduces the Avg Exec Time.

The resources of each server are fixed, so in order to shorten the Avg Exec Time, I hope to be able to perform multiple jobs on each server through the algorithm as much as possible to avoid waste of extra resources.

## Definition

* Fitness Value, it means that the current server has many available resources in addition to the resources required by the current job. There are three fitness values in ds-sim, namely fitnessCores, fitnessMemory, fitnessDiskSpace. If a specific fitness value is negative, it means that this resource of the current server does not meet the conditions for running the current job. Its calculation formula is:

Fitness Value = Specific Resource on Current Server - Specific Resource for Current Job required.

* SID, used to confirm the unique ID of each server, all subsequent operations will use SID as an identifier. Its calculation formula is:

SID = Server Type + Server ID.

* Weight, after calculating the fitness value of each resource of each server, it is put into a List in ascending order, and then the weight ratio of the specific resource of each server in this List is calculated. The closer the weight is to 0, the closer the fitness value is to 0, which means the closer the specific resources of this server are to the work requirements. Choosing a server with a lower weight to perform the current job can increase the resource utilization of this server. Its calculation formula is:

Weight = Index of Specific Resource List (asc sorted) / Specific Resource List Size.

* Total Weight, the total weight is to add up the weights of all resources of each server (in this case, the number of resources is three), and the total weight obtained is to find the server whose comprehensive resources are closest to the work requirements. Its calculation formula is:

Total Weight = Server Cores Weight + Server Memory Weight + Server Disk Weight.

* Parameter, a series of parameters used to evaluate the efficiency of the algorithm, including Actual Simulation End Time, Total Servers Used, Avg Utilisation, Total Cost, Avg Waiting Time, Avg Exec Time, Avg Turnaround Time.
* Performance Comparison, this parameter is for comparison with other algorithms (in this case, others are FF, BF and WF). Through the performance comparison, we can see which specific improvements or decreases in each algorithm of my algorithm. Its calculation formula is:

Performance Comparison = -((My Score / MIN(the Score of FF, BF, WF)) -1).

## Algorithm design

I will first obtain all the servers capable of the current job through the RESC Capable command and put them into a list. Then through for loop these servers, exclude servers whose server's state is unavailable or fitness value <0. I will put the rest of the server in the fitness map of various resources, map's key is SID, map's value is specific fitness value.

When all the servers have completed the for loop, my algorithm will check each fitness map. If there is any map is empty, it means that no current server configuration can meet the requirements of the current job, so I directly return the first server in the server list.

Next, I will put these fitness maps into the other 3 fitness lists, and sort them in ascending order according to their corresponding fitness value.

After the fitness list is sorted, my algorithm will start to calculate the fitness weight of the resources corresponding to each server.

Then, I will add the three fitness weights just calculated according to the SID of each server to get their total weight.

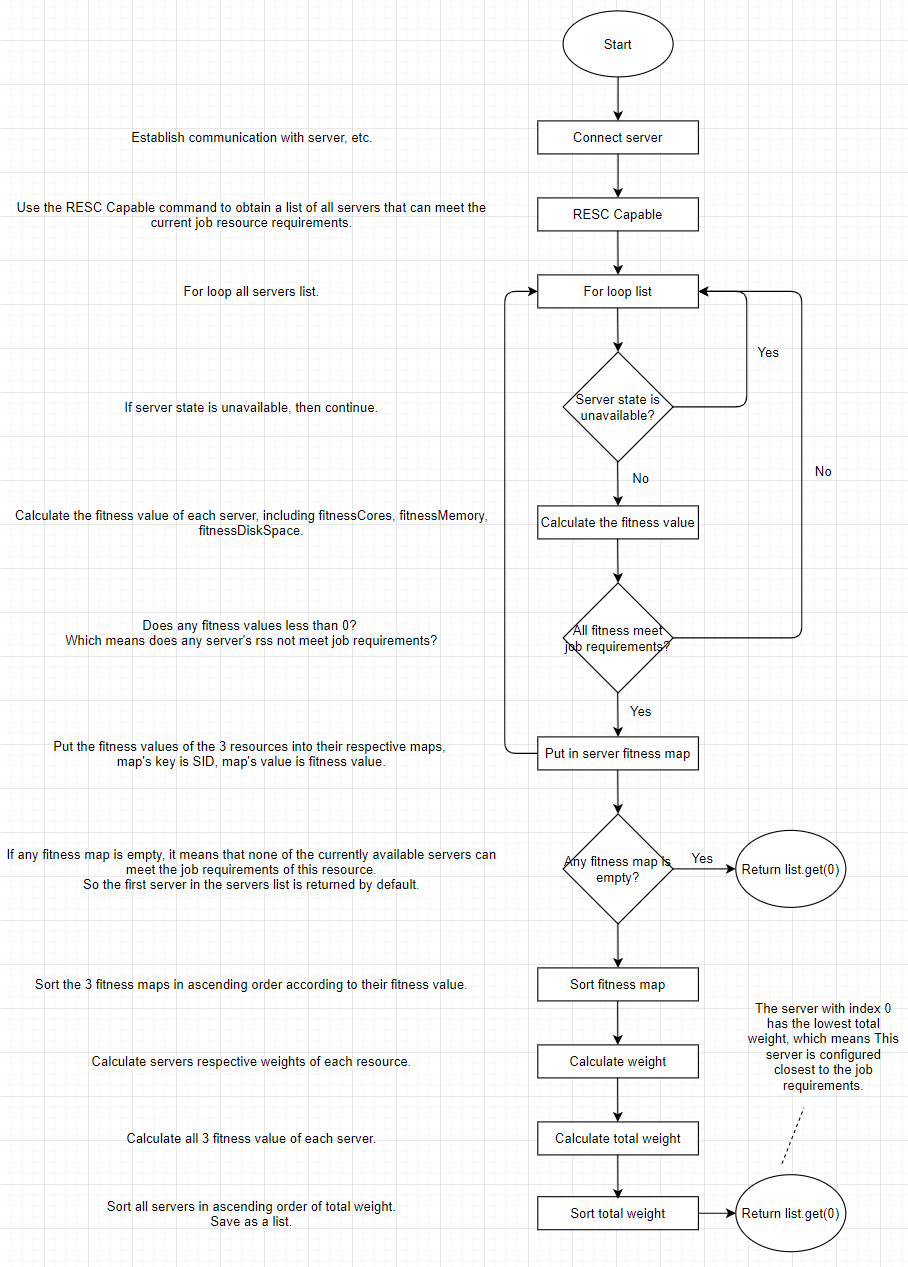
Finally, I also need to sort the total weight of each server in ascending order, and then return to the server with index 0. This is what I need to find the best server for the current job.

Figure 1 Algorithm Flowchart

## Pseudo code

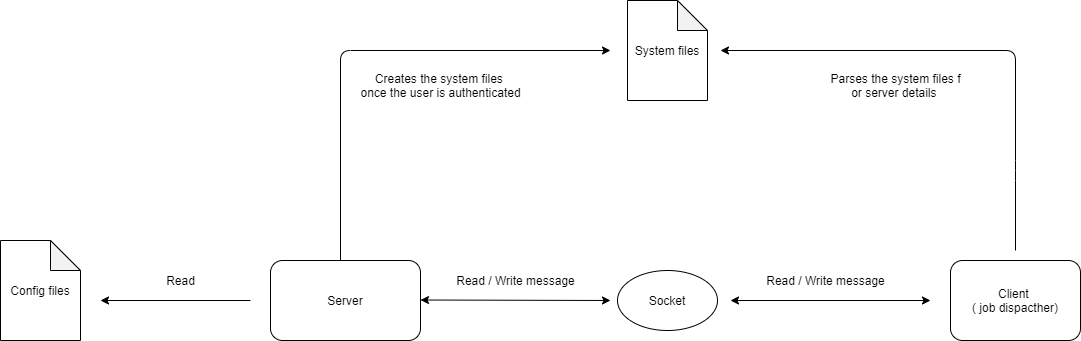


Figure 2 System architecture

## Performance Comparison

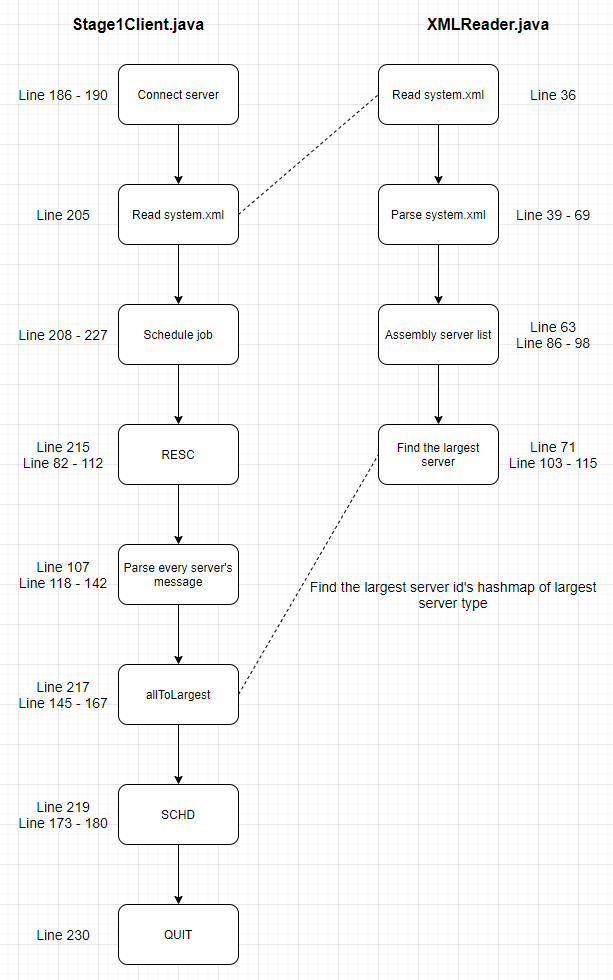


Figure 3 Implementation details

## References

* GitHub: <https://github.com/SnakeCN21/COMP3100-Group-Project/tree/S3_Snake>